第2章 作业管理与用户接口

作业的概念早于操作系统，虽然对于终端用户（如PC用户）来说已很淡化。然而在批处理系统中，用户是以典型的作业方式把任务交给计算机处理。目前，对作业的理解具有广义性，主要体现在宏观上或概念上。

教学要求

- 理解作业及其三种状态。
- 掌握对作业调度算法的几个评估公式：CPU利用率、吞吐量、作业平均周转时间和带权平均周转时间。
- 理解四个常见的作业调度算法：先来先服务、短作业优先、响应比高者优先和优先级法。
- 掌握系统调用与一般过程调用的区别。

教学内容

2.1 作业的概念

2.1.1 作业和作业步

2.1.2 作业的类型

2.1.1 作业和作业步

- 作业：用户交给计算机所做的工作的集合。
- 作业步：作业中的一个相对独立的步骤。如编程作业中的编辑、编译、连接、运行等几个作业步。

2.1.2 作业的类型

- 交互式作业（联机作业）：用户独占终端实施交互式控制，如用vi建立文件并进行修改等；
- 批处理作业（脱机作业）：作业由程序、数据、作业说明书三部分组成。程序是问题求解的算法描述；数据是程序加工的对象，但有些程序未必使用数据；作业说明书是告诉操作系统本作业的程序和数据按什么样的控制要求使之执行。

作业说明书示例（批处理文件）

- @echo off
- mkdir dos
- copy a.txt dos\b.txt
- type dos\b.txt
- echo ok!
- pause

2.2 作业管理的功能

2.2.1 作业的建立

2.2.2 作业控制块

2.2.3 作业的状态变迁

2.2.4 作业调度

2.2.1 作业的建立

主要方式：

- 联机输入：输入设备->主机
- 脱机输入：输入设备->磁带机->主机
- **SPOOLING**输入：

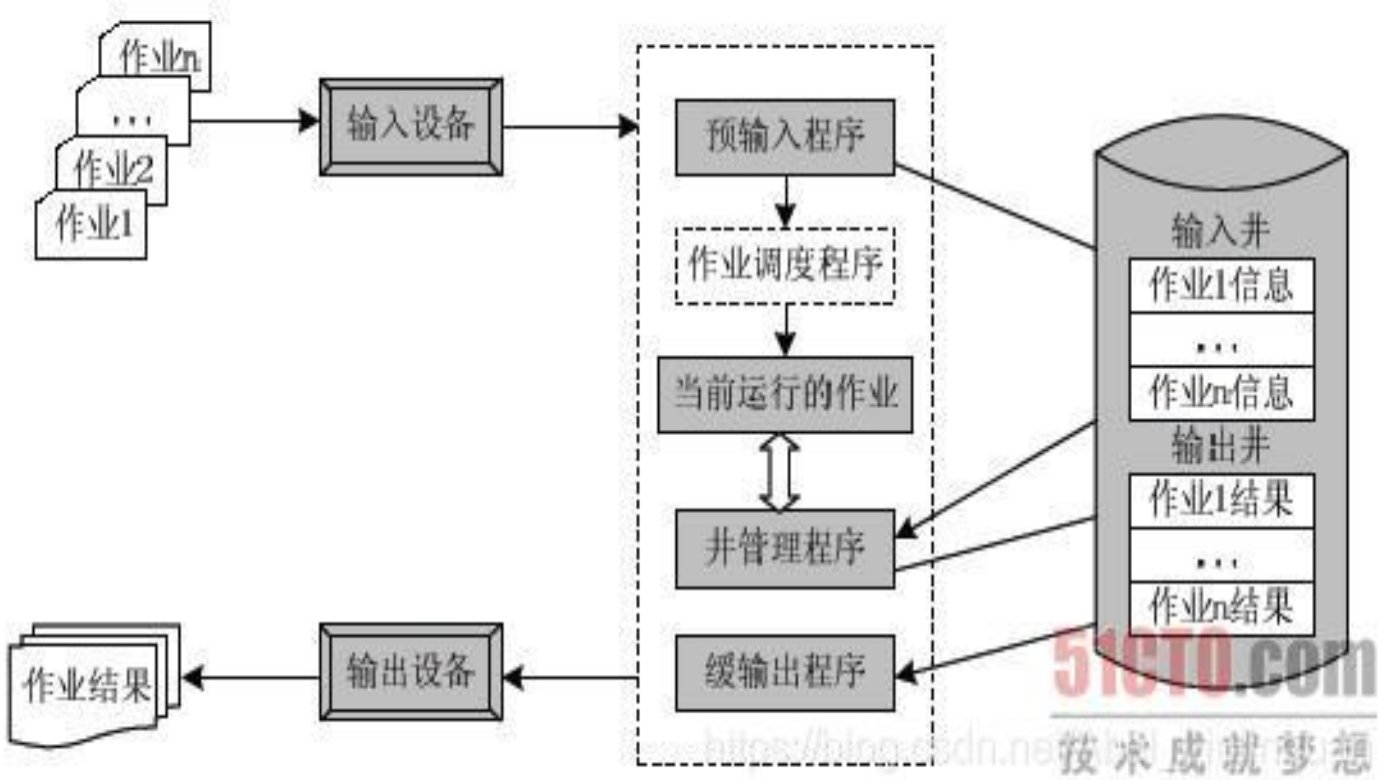
  外围设备实时相连，假脱机方式

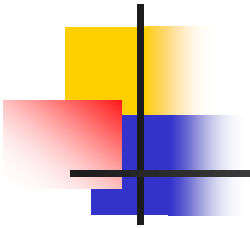
  输入设备->外存空间（输入井）->主机

  借助于通道（**I/O**处理机）实现

  特点：既可以联机修改，又具有脱机的快速特征

Spooling输入输出系统示意图





- SPOOLING输入方式是目前作业的主要输入方式
- 输入井中的多个作业形成一个作业后备队列
- 系统为每个作业建立一个作业控制块（JCB）

2.2.2 作业控制块

作业控制块(JCB)中包含了该作业的基本描述信息和控制信息，它是作业存在与否的唯一标志。包含内容如下：

- 描述信息。包括作业名、作业状态、作业的优先级和作业类型等。
- 资源要求。包括要求运行的时间、最迟结束时间、需要的主存空间、外设的种类和数量。
- 使用信息。包括作业进入系统的时间、开始运行时间、已经运行时间和内存地址等。

2.2.3 作业的状态变迁

- 收容状态（后备状态）。建立JCB，形成作业后备队列
- 执行状态。建立主进程，系统从作业管理转变为进程管理

  从微观来看，作业的执行状态可能是就绪、执行或等待（阻塞）三种状态中的一种。
- 完成状态。正常运行完成或因故障终止时

2.2.4 作业调度

1、 作业调度的概念

后备状态－>执行状态

2、作业调度的性能指标

（1）CPU利用率：CPU的工作效率；

（2）吞吐率：单位时间内完成的作业个数；

（3）平均周转时间：

周转时间=完成时间－提交时间

=等待时间＋运行时间

2.2.4 作业调度（续）

3、作业调度算法

- 先来先服务

  排队，简单公平，缺点是长作业阻塞短作业

- 短作业优先

  性能最佳，具有最短的平均周转时间

- 响应比高优先

  兼顾作业长短和到来次序

- 优先级高优先：

2.3 操作系统的用户接口

2.3.1 用户接口的功能

2.3.2 命令接口

2.3.3 程序接口

2.3.1 用户接口的功能和类型

- 功能：向用户提供使用计算机的接口
- 类型：**命令接口和程序接口**

2.3.2 命令接口

1、联机命令接口

由键盘命令和屏幕命令组成。

- 键盘命令是由联机用户在交互式终端上通过键盘键入的命令，体现人机之间的交互性。
- 屏幕命令也就是图形化命令。它由窗口、图标、菜单、对话框等图形化元素构成。

2、脱机命令接口

- 作业控制卡方式
- 作业说明书方式

2.3.3 程序接口

程序接口是向编程人员提供的，由一系列系统调用（**System Call**）指令组成。

1、系统调用的概念

系统调用是指在用户程序中，调用了操作系统中能完成某些特定功能的例行程序，即用户程序对OS的调用。例如打印、读写磁盘等工作；

系统调用实例（汇编程序员观点）

- 程序段1：利用系统调用打印（用INT指令调用DOS的功能来完成）

```
MOV AH,05H
MOV DL，Char
INT 21H
…
Char DB '1'
```

当INT返回时，打印已经完成

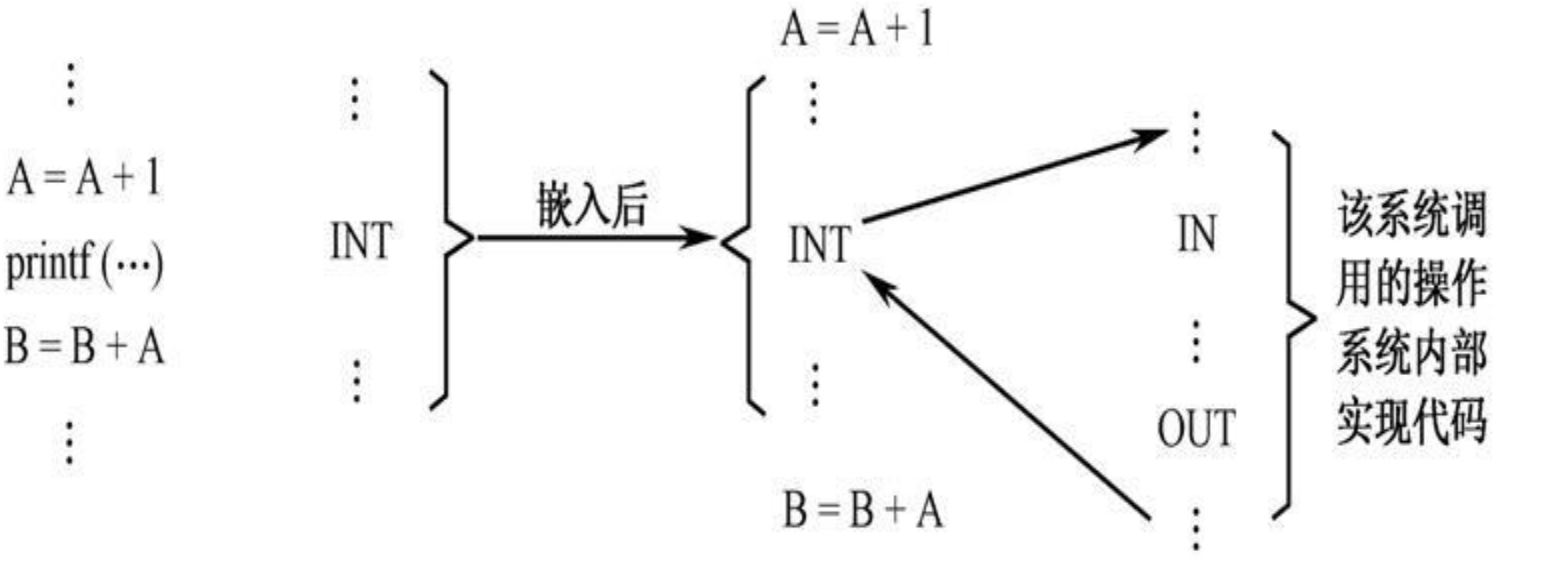
- 程序段2:不用系统调用，而用IN/OUT指令直接读写打印机的接口寄存器

```
      MOVE I,0
L1:MOVE I,I+1
      CMP I,5
      JNC L3          打完五个字符
      MOVE A,I
L2:IN ADDR1,B         状态寄存器
      OR B,BS         状态位选择码
      JNC L2          未准备好
      OUT ADDR2,A     打印,数据寄存器
      JMP L1
L3:RET
```

为什么我很少接触系统调用?

- 通常仅汇编程序员才会接触到系统调用，而高级语言程序员通常接触不到系统调用，只接触库函数。
- 通过库函数来实现对操作系统的“间接”调用。

高级语言中对操作系统的“间接”调用



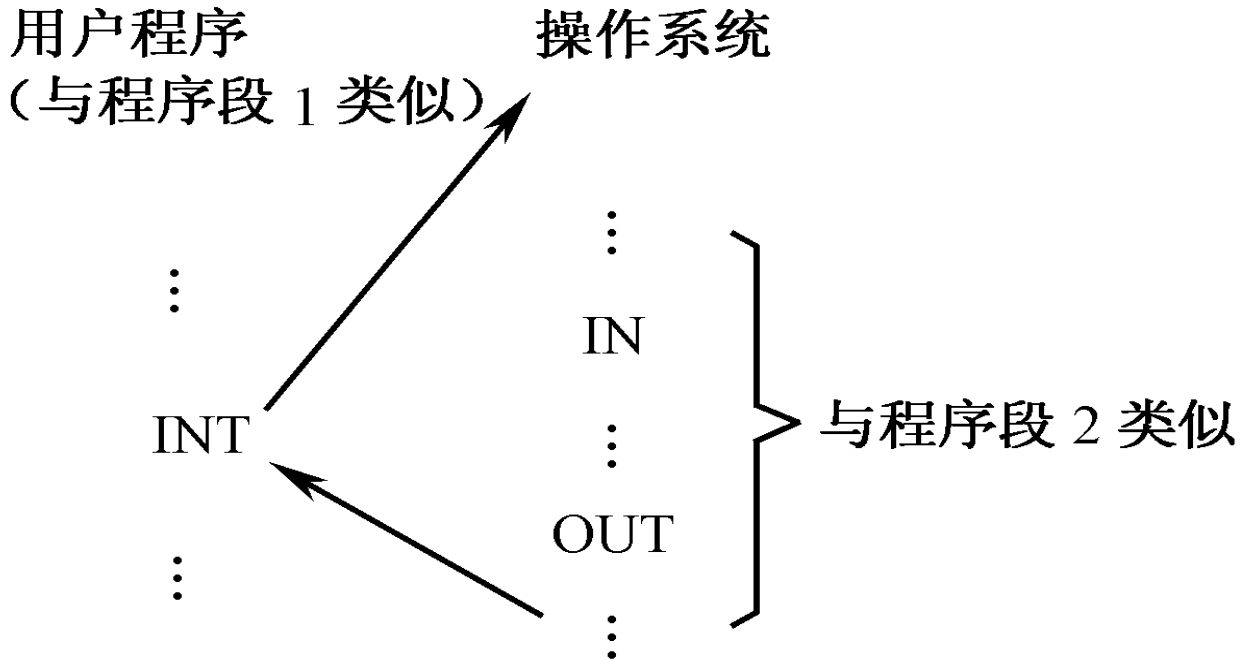
(a) 源程序段　(b) printf 目标代码（平时包含在函数库中，在用户程序编译时嵌入用户程序）　(c) 程序段　编译产生的可执行目标程序（函数已嵌入）　(d) 操作系统内部代码

2、系统调用指令的实现及其特点

（1）机器、OS与系统调用指令间的关系

- 每种OS提供几十至几百个系统调用
- 每种机器仅提供一个系统调用指令：
  - 例:SUN－TRAP指令
  - SGI工作站－SYSCALL指令
  - IBM PC－INT指令

（2）系统调用指令的实现机制（借助中断机制）



（3）系统调用指令特点

- 系统调用指令由机器（CPU）提供，而其调用的功能由OS提供；
- 不同的系统调用使用同一条系统调用指令，但指令参数(功能号或寄存器)不同；

2.3.3 程序接口（续）

2、系统调用的分类

- 文件管理类。如创建文件、打开文件、读写文件、关闭文件等；
- 进程管理类。如进程创建、撤消、唤醒以及进程间通信等；
- 系统管理类。如取日历时间、取或设置终端信息等；
- 设备管理类。

3、系统调用与一般过程调用的区别

- 执行状态不同

  前者调用方在用户态，被调用方在核心态，后者调用方与被调用方同属一个状态，核心态或用户态。

- 执行方式与过程不同

  前者需使用软中断指令（**int**）或陷入指令（**trap**）；后者使用普通的跳转指令（**call**、**jmp**等）

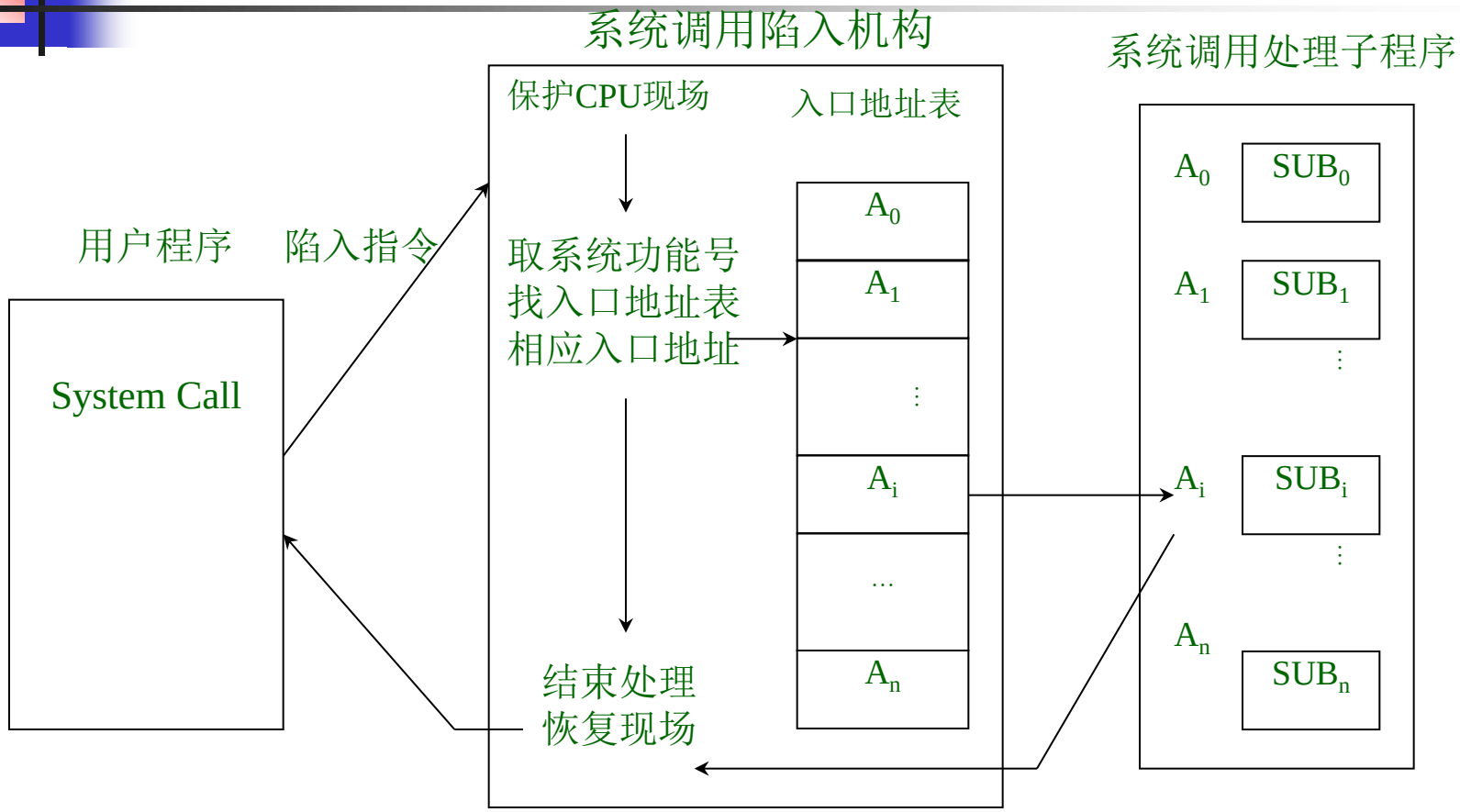
- 提供的方式不同

  前者由**OS**提供，后者由编译系统提供

- 执行的代码不同

  前者执行是**OS**内核代码，后者执行的是用户自己编写的代码

4、系统调用的处理过程


系统调用陷入机构
系统调用处理子程序
保护CPU现场
入口地址表
用户程序
陷入指令
System Call
取系统功能号
找入口地址表
相应入口地址
结束处理
恢复现场
A0
A1
Ai
An
SUB0
SUB1
SUBi
SUBn

系统调用与库函数

- 在程序设计语言(如C语言)中，往往提供与各系统调用对应的库函数，应用程序可通过对应的库函数来使用系统调用。
- 库函数的目的是隐藏访管指令细节，使系统调用更象过程调用，但一般地说，库函数属于用户程序而非系统程序。
- 操作系统为用户提供系统调用也出于安全和效率考虑，使得用户态程序不能自由地访问内核关键数据结构或直接访问硬件资源。

【返回】

2.4 Windows的API

Windows支持API的三个组件：

- **Kernel**包含了多数操作系统函数，如内存管理、进程管理；
- **User**集中了窗口管理函数，如窗口创建、撤销、移动、对话等相关函数；
- **GDI**提供画图函数、打印函数。

Windows将三个组件置于动态链接库**DLL**中。

Win32API和Linux系统调用粗略对应关系

UNIX/Linux	Win32	说明
fork	CreatProcess	创建进程
waitpid	WaitForSingleObject	等待进程终止
open/close	CreatFile/CloseHandle	创建或打开/关闭文件
read/write	ReadFile/WriteFile	读/写文件
lseek	SetFilePointer	移动文件指针
mkdir/rmdir	Creat/Remove Directory	建立/删除目录
stat	GetFileAttributesEx	获得文件属性

2.5 Linux的用户接口

2.5.1 Linux的Shell

功能：

- 命令行解释
- 通配符
- 输入/输出重定向和管道
- **Shell环境控制**
- **Shell编程**

2.5.2 Linux的系统调用

1. Linux系统调用的类型

- 设备管理类
- 进程控制类
- 文件系统类
- 存储管理类
- 进程通信类

2. Linux系统调用的实现

系统调用控制程序的功能为：

- 获得系统调用号，检验其调用的合法性；
- 建立调用堆栈，保护**CPU**现场信息；
- 根据系统调用号定位内核函数地址；
- 根据通用寄存器内容，从用户栈中取入口参数；
- 执行内核函数，返回结果给应用程序；
- 执行出栈操作，判断调度程序**scheduler**是否要被执行。

3、库函数与系统调用的分层关系